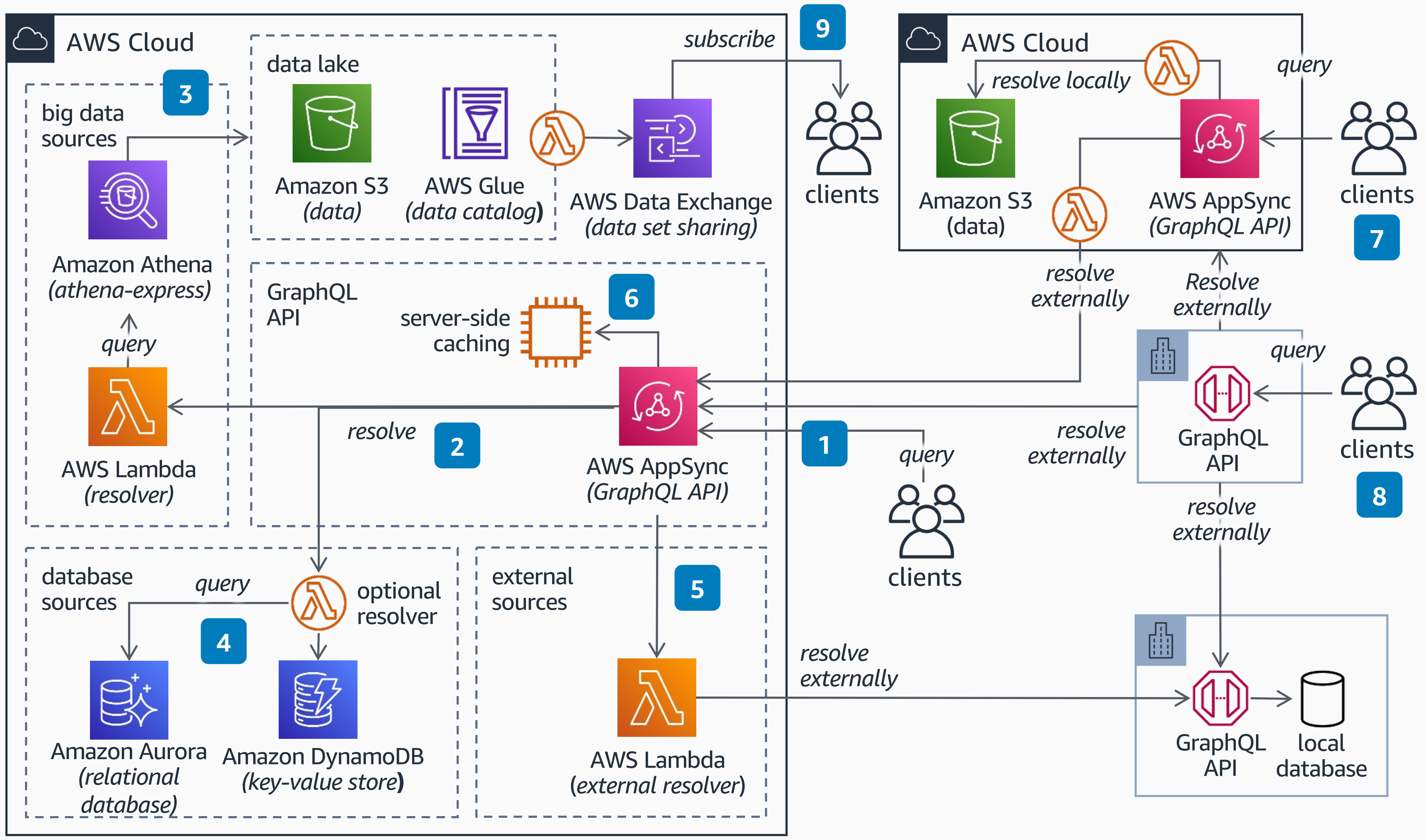
# Synchronous Data Mesh for GraphQL queries

Use an API composition pattern to build a modern, distributed, and decentralized data architecture. It enables clients to query data where it lives, without first transporting it to a data lake or data warehouse. It also allows domain-specific teams to own and serve data as a product.



1 Expose your domain's schema via an HTTPS API with **AWS AppSync**, allowing users to dynamically query a domain's data with GraphQL syntax.

2 Compose the returning query results with a combination of resolvers built with **AWS Lambda** functions.

3 For data partially available in a data lake, retrieve the data by running SQL queries supported by **Amazon Athena**, running the **Athena** jobs asynchronously with the athena-express library.

4 For data partially available in databases, retrieve the data either directly from **AWS AppSync**, or by using **Lambda** functions as proxy resolvers to your databases.

5 For data partially available in external sources, use **Lambda** resolvers to fetch the data by invoking remote HTTP APIs.

6 To improve your API's performance, enable server-side caching on **AWS AppSync**.

7 Other parties using AWS can replicate this architecture in their domains, also allowing clients to query their APIs using GraphQL, and composing the results with internal and external resolvers – including your domain's GraphQL API.

8 Parties that don't AWS can also expose their domains with GraphQL APIs built with other technologies. This provides a seamless experience to clients, while composing query results with data sets resolved from multiple external APIs.

9 To retrieve large datasets, clients can subscribe to **AWS Data Exchange** instead.

*Reviewed for technical accuracy April 25, 2022*

AWS Reference Architecture